himalayan linguistics

A free refereed web journal and archive devoted to the study of the languages of the Himalayas

Himalayan Linguistics

Segmenting and POS tagging Classical Tibetan using a Memory-Based Tagger

Marieke Meelen

University of Cambridge

Nathan W. Hill

SOAS, University of London

ABSTRACT

This paper presents a new approach to two challenging NLP tasks in Classical Tibetan: word segmentation and Part-of-Speech (POS) tagging. We demonstrate how both these problems can be approached in the same way, by generating a memory-based tagger that assigns 1) segmentation tags and 2) POS tags to a test corpus consisting of unsegmented lines of Tibetan characters. We propose a three-stage workflow and evaluate the results of both the segmenting and the POS tagging tasks. We argue that the Memory-Based Tagger (MBT) and the proposed workflow not only provide an adequate solution to these NLP challenges, they are also highly efficient tools for building larger annotated corpora of Tibetan.

KEYWORDS

Tibetan, word segmentation, Memory-Based Tagging

This is a contribution from *Himalayan Linguistics, Vol. 16(2): 64-89.*
ISSN 1544-7502

Himalayan Linguistics, Vol. 16(2). © Himalayan Linguistics 2017
ISSN 1544-7502

Segmenting and POS tagging Classical Tibetan using a Memory-Based Tagger[1]

Marieke Meelen
University of Cambridge

Nathan W. Hill
SOAS, University of London

1 Introduction

The ultimate goal of this research is to create a large corpus of Classical Tibetan that is properly segmented and provided with Part-of-Speech (POS) tags.[2] The process of POS tagging consists of assigning a morphosyntactic class to each word in a text. The words are automatically tagged on the basis of their characteristics and of the context in which they occur. Fully annotated linguistic corpora consist of POS tags, information-structural and syntactic annotation. These searchable corpora are indispensable tools for researchers in Tibetan studies, because they provide access to a wealth of systematically organised data for anyone interested in philology, language and Tibeto-Burman linguistics. In addition, scholars in theoretical linguistics and Natural Language Processing will benefit from consistently annotated corpora that facilitate their research in comparative linguistics and language typology. Fully annotated corpora in a well-designed database allow for queries of lexical items, morphosyntactic categories, collocations, orthographic and morphological variation, word order, syntactic patterns, etc.

In this paper we present the results of a memory-based POS tagger assigning highly detailed morphosyntactic tags to a Classical Tibetan corpus. Classical Tibetan texts usually consist of long strings of characters that are only delimited by a phrase-final ། (‘*śad*’) or topic-final ། (‘*ñis śad*’). An automatic POS tagger, however, assigns a tag to each word and therefore only works if word boundaries are clearly indicated. When word boundaries are indicated by spaces, the necessary task of word segmentation (finding word boundaries in a corpus) is easily done. In languages with writing systems like Chinese, Japanese or Classical Tibetan in which words are not (always) indicated by spaces, word segmentation presents a challenging task in Natural Language Processing (NLP) that has to be solved *before* morphosyntactic annotation can take place.

Following Zhao et al. (2006) and Huidan et al. (2011), we reformulate Classical Tibetan word segmentation as a syllable tagging problem. We propose that syllable tagging can be approached in

1 This research has benefited from the generosity of the European Research Council under the auspices of the Synergy Grant “Beyond Boundaries: Religion, Region, Language and the State” (ID 609823) and the Advanced Grant “Rethinking Comparative Syntax” (ID 269752).

2 See Meelen, Hill and Handy (2017a) for a preliminary segmented version and (2017b) for a preliminary POS-tagged version of this Classical Tibetan corpus.

the same way as POS tagging and, furthermore, that a memory-based tagger is a suitable tool for Tibetan word segmentation and the subsequent POS tagging, because it renders robust results with a minimum amount of preprocessing.[3] In the next section we introduce the corpus material and certain issues concerning the Tibetan language that are relevant to the current NLP tasks. Section 3 describes the methodology of the three-part workflow consisting of memory-based syllable tagging, rule-based conversion into words (which combined solve the word segmentation problem) and memory-based POS tagging. Finally, in section 4 we evaluate the results.

2 Corpus material

This study uses the Classical Tibetan corpus produced by the project "Tibetan in Digital Communication." This corpus includes the *Mdzaṅs blun* (9th century, canonical), the *Bu ston chos ḥbyuṅ* (13th century, ecclesiastical history), the *Mi la ras paḥi rnam thar* and *Mar paḥi rnam thar* (15th century, biography).[4] This corpus is in no way balanced and cannot be seen as particularly representative. The choice was dictated by the availability of digital texts that are well studied and understood. These texts have been popular in the study of Tibetan grammar and in the teaching of Classical Tibetan. All of the works included in the corpus are primarily prose, but contain short poetic passages. The same corpus is used for both segmentation and POS tagging.

3 Methodology

The workflow we present here in order to provide Classical Tibetan texts with highly detailed morphosyntactic annotation consists of three main steps after initial preprocessing illustrated in Figure 1:

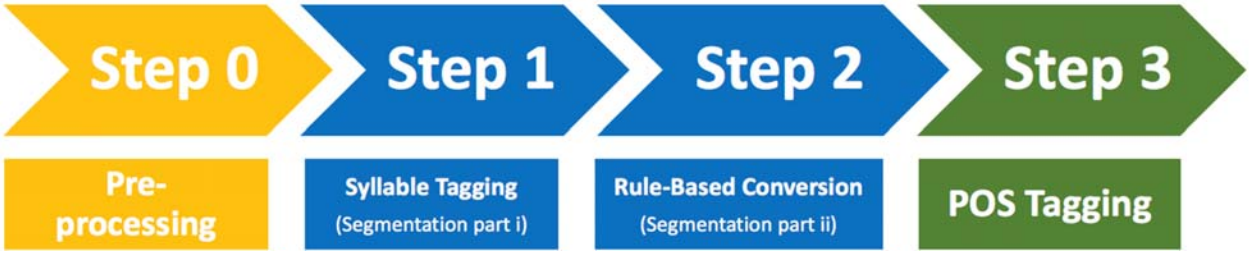

Figure 1. Workflow for a POS-tagged corpus of Classical Tibetan

In this section we briefly discuss the morphosyntactic tag set and certain textual conventions, as well as the background of our training data. We then outline the workflow, starting with some minimal preprocessing, followed by word segmentation (syllable tagging + rule-based conversion) and, finally, POS tagging.

3 The aim of the current paper is merely to present the proposed workflow and show how our ultimate goal of creating a large annotated corpus of Classical Tibetan is achieved by using a Memory-Based Tagger because the minimal amount of preprocessing that is required for this tagger makes it quick and easy to use. Other types of automatic taggers, such as Conditional Random Field or Neural Network taggers may yield slightly better results, but we leave a detailed comparison of accuracies of different taggers to future research.

4 For a list of previous studies of the *Mdzaṅs blun* and *Mi la ras paḥi rnam thar* see Hill 2012: 10.

3.1 Textual conventions and normalisations

The above-described corpora were normalised to a uniform type of unicode, because in some cases there are two ways to encode a Tibetan character. The following changes were made:
- "།" (Unicode 0f0e) replaced by "།" (Unicode 0f0d+0f0d)
- "་" (Unicode 0f0c) typical after ང and before ། changed to "་" (Unicode 0f0b)

3.2 Development of the training sets

The training data used to test the new approach to word segmenting and POS tagging in the present paper was developed by the collaborators of the "Tibetan in Digital Communication" project at SOAS, University of London (cf. www.soas.ac.uk/cia/tibetanstudies/tibetan-in-digital-communications/). It consists of two separate sets: one for syllable tagging and one for POS tagging. The training set for syllable tagging consisting of a Tibetan syllable + syllable tag. For the present paper we tested the manually corrected training sets consisting of >377*k* correctly identified syllables with two slightly different tag sets to test the results for both (see section 3.4 below for details on the tag sets).

The training set for morphosyntactic tagging consists of >318*k* word tokens of Classical Tibetan in total. Garrett et al. (2014) explain how they first manually tagged >17*k* words of the *Mdzaṅs blun*, then used various Tibetan lexicons to create a lexical tagger that assigned all possible tags to a further 27*k* words of the same *Mdzaṅs blun* and the first 32*k* words of the *Mi la ras paḥi rnam thar*. Words with multiple tags were subsequently disambiguated by an elaborate set of rules, described in detail in Garrett et al. (2015). This rule-based tagger in addition provided suggestions aiding further manual tagging and correction of the training data. Further details about developing the training data can by found in Garrett et al. (2015).

3.3 Step 0: Pre-processing

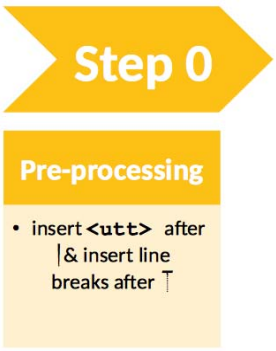

Figure 2. Step 0 of the workflow

A major advantage of the workflow with the Memory-Based Tagger proposed in the present paper is that future data need very limited preprocessing. If the new data are already split up into Tibetan word tokens, the first two stages (syllable tagging and rule-based conversion) can be skipped and a simple regular expression converting spaces into line breaks and the insertion of utterance boundaries suffice to provide the right format as input for the tagger.

As indicated in the introduction, however, most Tibetan texts consist of continuous strings of characters, only delimited by a sentence-final ། ('*śad*') or topic-final ། ('*ñis śad*'). Utterance boundaries that are necessary for the tagger can easily be inserted after these delimiting characters, but since there are no white spaces for word boundaries, tokenisation is a challenging task. Following Zhao et al (2006) and Huidan et al. (2011), we propose to approach this as a syllable tagging problem. Syllables in Tibetan are often delimited by a punctuation mark that looks like a high full stop: · *tsheg*. Tibetan texts with continuous strings

of characters can thus automatically be tokenised by inserting spaces following the *tsheg* marker. These tokenised syllables marked by *tsheg* then each get a "syllable tag" in the first step of our workflow described in the following section. There is no further lemmatisation, stemming or regularisation of any kind (e.g. orthography) necessary apart from these insertions of syllable and utterance boundaries.[5]

3.4 Step 1: Syllable tagging (part i of word segmentation)

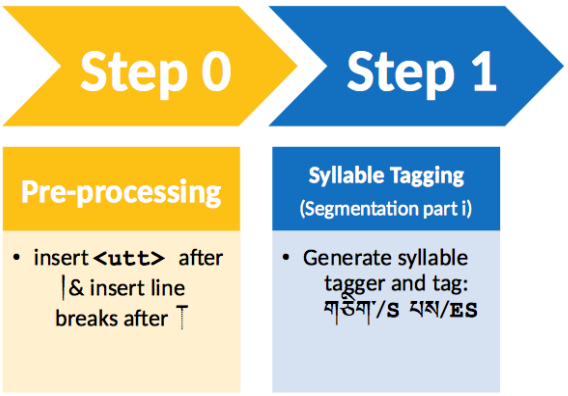

Figure 3. Steps 0 and 1 in the workflow

After tokenisation of the Tibetan text into syllables, we used the Memory-Based Tagger generator (MBTg) based on the TiMBL software package designed by Daelemans et al. (1996).[6] We chose MBT because it is freely available, easy to train on a new language and because we expected the model to behave robustly in the face of (orthographic) variation.

Each token (i.e. a syllable delimited by the Tibetan marker *tsheg*) was given a tag describing its position with respect to Tibetan words. Following Huidan et al. (2011), syllables occurring in the beginning, middle or end of a Tibetan word are tagged as "B," "M" or "E." A syllable can also be a word in itself, in which case it will be tagged "S." Finally, a token can be identified as a combined location "ES" or "SS" for two morphemes that are joined together in a single syllable, e.g. འདིའི *'di'i* 'of this' (to be tagged འདི་/SS and ultimately segmented འདི་ འི་) or ཆེན་པོས་ *chen pos* 'by the great' (to be tagged ཆེན་/X པོས་/ES and ultimately segmented ཆེན་པོ་ ས་). Garrett et al. (2015) further extended the syllable tag set by splitting up the category indicating the middle of the word (M) into "Y," "Z," and "M," where "Y" is the second non-terminal syllable of a word, "Z" is the third non-terminal syllable, and "M" is the fourth or later non-terminal syllable. For the present paper we tested the memory-based tagger on the small syllable tag set (consisting of B, M, E, S, ES and SS) and the larger tag set in which B appears as X and M is split up into Y, Z, and M. These tags are illustrated in Table 1; Figure 3 shows this next step in our workflow.

5 Note that if texts with further markup, e.g. electronic texts with XML-markup, are used, this additional markup will have to be stripped before tagging. Line and page numbers could be treated as separate tokens to facilitate subsequent conversion back to a proper XML format for various applications. We will discuss this in future work.

6 This is freely available from https://languagemachines.github.io/mbt/. A detailed user manual and reference guide was published by Daelemans et al. (2010) and further details on the principles and applications of memory-based learning can be found in Daelemans and Van den Bosch (2005).

Small tag set (Huidan et al. 2011)		**Larger tag set (Garrett et al. 2015)**		**Example**
S	single syllable	S	single syllable	གཅིག་ *gcig* ‘one’
B	beginning of the word	X	beginning of the word	བྱང་/X ཆུབ་/Y སེམས་/Z དཔའ་/E → བྱང་ཆུབ་སེམས་དཔའ་ *byaṅ chub sems dpa’* ‘Bodhisattva’
M	middle of the word	Y	beginning, following X	
M	middle of the word	Z	beginning, following X–Y	
E	end of the word	E	end of the word	
M	middle of the word	M	middle of the word	ཀ་/X ན་/Y ཤི་/Z ནི་/M པ་/M ལི་/E → ཀ་ན་ཤི་ནི་པ་ལི་ *ka na śi ni pa li* (name)
ES	end + single syllable	ES	end + single syllable	པས་ *pas* ‘because, by’
SS	two single syllables	SS	two single syllables	དཔེར་ *dper* ‘for example’

Table 1. Syllable tag sets with examples

MBT uses a Nearest-Neighbour classifier to assign a tag to a word in the test set. It does so on the basis of a labeled training set. From this training set, it generates a tagger. First, the model extracts features, such as initial and final graphemes of tokens and other tokens and tags in the preceding or following context. It then uses this knowledge to find the most closely resembling tag, using a small set of features if it has seen the target word or syllable in the training set (known tokens), and backing off to a bigger set in case the target word or syllable has not been observed before (unknown tokens). MBT was originally designed to assign morphosyntactic tags to words. For the present paper, however, we have attempted to use it to identify syllables and thus solve the first part of the challenging word segmentation task in Tibetan.

3.4.1 Generating a memory-based Tibetan syllable tagger

At first, the tagger was created on the based of a training set with the standard parameter settings of MBT. The parameter settings can be adjusted for both known and unknown tokens, according to what works best for a particular corpus (see Appendix A). Known tokens are those that are found in the exact same form in the training set and are thus easier to classify than unknown tokens. Many syllables in Tibetan can occur in several places in a word and can therefore receive different tags. བླུན་ *blun* ‘to be dull’, for example, can occur on itself and thus be tagged “S,” but it can also occur at the beginning (e.g. བླུན་པོ་ *blun-po* ‘foolish’) or end of the word (e.g. མཛའ་བླུན་ *mdzaṅs blun* ‘the wise and the foolish’) and thus be tagged “B/X” or “E” respectively. With specific parameter settings, we made the tagger use the tokens and tags in the immediately preceding and following context to disambiguate these cases.

Tokens that do not occur in the training set are much more difficult to classify. In these cases, the tagger not only uses the preceding and following context, but it can also identify the first or last characters of a token. In languages with morphological endings, for example, the tagger can learn to

recognise certain verb forms if they have a similar suffix. The English suffix *-ed* in words like *worked, danced, listened* that are tagged as past-tense verbs will be recognised in tokens the tagger has not yet encountered, such as *climbed* or *experienced*, and tag those as past-tense verbs as well. The results for a certain set of parameter settings for a corpus can be evaluated with a 10-fold cross-validation (see the results in section 4 below).

One of the advantages (alongside good results with a minimal amount of necessary preprocessing) of using the Memory-Based Tagger generator (MBTg) is that the user does not need to understand how the algorithm or parameter settings work exactly. The only required input is a training corpus consisting of a set of manually annotated (and/or corrected) tokens + tags. The MBTg then automatically generates the tagger in the form of a settings file. In this way, thousands of words can be tagged per second. Unlike statistical taggers that use Hidden Markov Models (HMM-taggers), there is no need for any additional smoothing for sparse data since this is already built into the similarity-based model (cf. Zavrel and Daelemans (1997)). Spelling, morphology, context and the syllables or words themselves are all sources of information integrated in the weighted similarity metric.

3.4.2 Tagging syllables with the generated tagger

The tagger generator creates data files (e.g. a list of the most frequent tokens + tags) and a settings file that defines the Memory-Based Tagger (MBT) for Tibetan syllables, in this particular case. This settings file is used to assign tags to a new part of the corpus. This new corpus should be presented to the tagger as a tokenised text file with utterance boundaries that mark the end of a sentence (or part of a sentence, if preferred). This format is shown in Figure 4a (before preprocessing) and Figure 4b (after preprocessing):

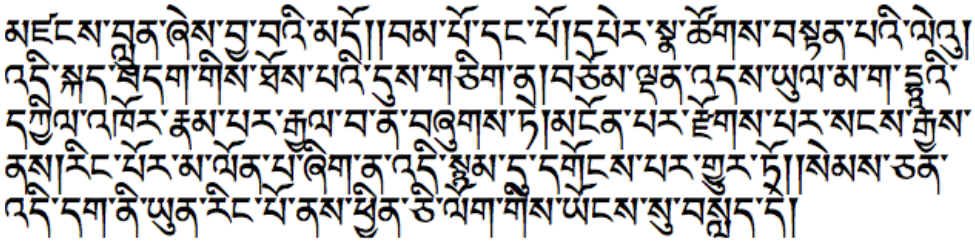

Figure 4a. Sample of unsegmented Tibetan input

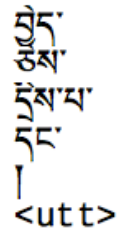

Figure 4b. After preprocessing

Based on the word lists derived from the training set, the MBT then divides the new text in need of annotation into 'known' and 'unknown' tokens. After this, it assigns a tag to each word based on the chosen parameter settings. The results can be written into a text file, as shown in Figure 5. The tokens are followed by a / when it is a "known" token, i.e. when it occurred in the most frequently found tokens in the training set. Tokens not found in this list are followed by a double // and then the tag that informs us about the location of the syllable in the Tibetan word.

```
དེས/SS དེ/S ལ/S དགའ/S མགུར/S སྒྲ/X བའི/ES ཐྲིར/SS གོས/S དང/S |/S <utt>
ཀུན/S རྣམས//S བསྐར/X བ/E དང/S |/S <utt>
དེས/SS གོས/S དང/S ཀུན/S རྣམས/S བགོས//S ནས/S སོང/X བ/E ལས/S |/S <utt>
སྐུས/X པ/E གཞན/S ཞིག/S ཀྱང/S དོན/S ཚེ/S ཕྱ/S བཏུ/S ཐོགས/S ཏེ/S འོངས/S ནས/S |/S <utt>
བཟང/X མོ/E དགའ/X མགུར/E སྒྲ/S ཀྱིས/S ཚུར/SS ཤོག/S ཅེས/S སྨས/S སོ/S |/S <utt>
དེས/SS བསམས/X པ/E |/S <utt>
```

Figure 5. Sample of the syllable tagger output

This way of syllable tagging with limited preprocessing takes a minimum amount of time and effort and could thus easily be extended to new and much larger corpora. Although statistical Part-of-Speech tagging never achieves 100% accurate results, the output is sufficiently accurate to use as a starting point to create a large annotated corpus of Tibetan. In section 4.3 we discuss the evaluation of the syllable tagger in more detail.

3.5 Step 2: Syllable tagging (part ii of word segmentation)

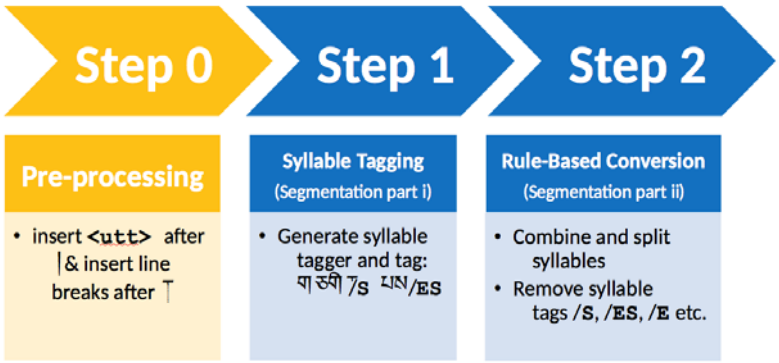

Figure 6. Steps 0, 1, and 2 in the workflow

The output of the syllable tagger then needs to be converted to actual Tibetan words. This can be done by designing rules that combine or separate syllables based on the assigned tag. The tag "S" for lone syllables can simply be removed to achieve the desired result, but the syllables with other tags require further rules in order to combine (or split) them into proper Tibetan words. Multisyllabic words can for example consist of syllables that are tagged "X" (beginning) + "E" (end), "X" (beginning) + "M" (middle) + "E" (end) or combinations and extensions of those, e.g. X-Y-E, X-Y-Z-E or X-Y-Z-M-M-E, etc. In addition, there are two tags for complex syllables in Tibetan, i.e. syllables that are not separated by a *tsheg* marker but should be separated in segmenting. An example of this is, འདི་ *'di'i* 'of this', tagged "SS," that should for the purposes of later POS tagging be separated into two elements: འདི + འི་. In a similar fashion, "ES" is used for a word-final syllable that needs to be joined together with the following single syllable word, e.g. ཡིས་ *pos* 'agent + instrumental case' (tagged "ES") joined with ཆེན་ *chen* 'great' (tagged "X") to get ཆེན་ཡི + ས *chen po-s* 'by the great'. We developed a python script[7] that automatically converts the syllable-tagged files into proper Tibetan words based on the set of rules listed in Table 2:

7 The script is available under a creative commons licence on https://github.com/tibetan-nlp.

Replacement rules
Attach /X, /Y, /Z, /M to their following words.
Attach /E to its preceding word, leaving a space.
Isolate tokens with /S as single words.
Add a space before and after attached end syllables: ར་, འི་, ས་ and འོ་ + /ES
Add a space before and after attached single syllables: ར་, འི་, ས་ and འོ་ + /SS
Add a space before and after attached end syllables without *tsheg*: ར, འི, ས and འོ + /ES
Add a space before and after attached end syllables without *tsheg*: ར, འི, ས and འོ + /SS
Add a space before and after attached end syllables with/without *tsheg*: འམ, འམ་, འང and འང་ + /ES
Add a space before and after attached end syllables with/without *tsheg*: འམ, འམ་, འང and འང་ + /SS
Delete all tags (and, depending on desired output, also utterance boundaries <utt>[8])

Table 2. Overview of all replacement rules

The script converts one or multiple file(s) with tagged syllables and creates a time-stamped folder with new output files. Samples of in- and output are shown in Figures 7a and 7b:

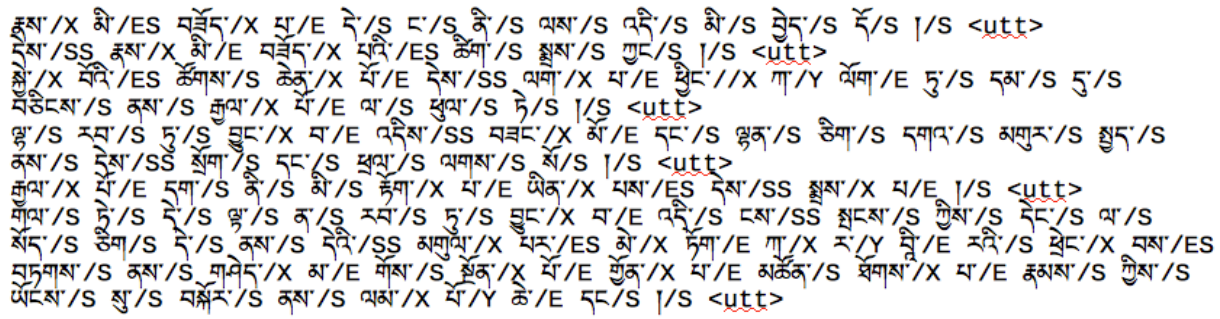

Figure 7a. Sample of input file (output of the syllable tagger)

Figure 7b. Sample output file (result of segmentation into Tibetan words in two parts)

8 For the present workflow, utterance boundaries in the form of <utt> need to be present in the output file of the word segmentation stage as well so that these output files can directly be used as input for the subsequent stage of POS tagging.

The tags ES and SS that are automatically assigned by the tagger are usually mistakes. In rule-based conversion, these will be corrected and treated as E and S respectively. Only ར་, ིའ་, ས་, ོའ་, འམ་, and འང་ are able to be written orthographically as part of a preceding syllable. Thus, only tokens ending with ར་, ིའ་, ས་, ོའ་, འམ་, or འང་ may take the tags ES and SS. If the system tags any other tokens with these tags, we know that it is in error. The reason for converting respectively to E and S, is simply to defer to whatever wisdom the machine may have shown. In other words, we know it is wrong about the second S of the tag, but maybe it is right about the initial E or S.

3.6 Step 3: POS tagging

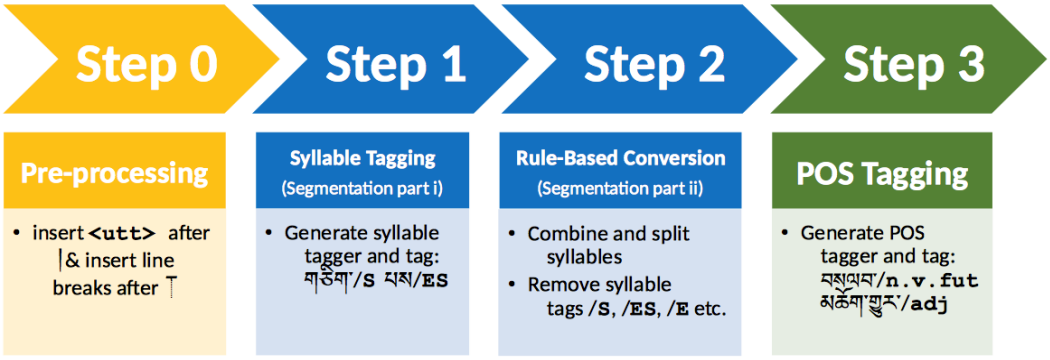

Figure 8. Complete workflow to create a large annotated corpus of Classical Tibetan

The morphosyntactic tag set used for the present paper consists of 79 tags denoting the morphosyntactic features of the Tibetan word forms. The tag set is based on the set developed by Garrett et al. (2014), described in further detail in Garrett et al. (2015). There are some slight differences between the current tag set and those in Garrett et al. (2015), listed in Table 3:

Replaced Tags	Denotation
`cl.tsam` > `d.tsam`	clitics *-tsam, -sñed, -sñad*
`cl.emph` > `d.emph`	emphatic noun phrase clitics
`cl.indef` > `d.indef`	indefinite markers
`n.v.redup` > (following tag)	Reduplication

Further specified	Denotation
`v`	verb

Additional Tags	Denotation
`adv.mim`	mimetic adverbs
`case.nare`	case marker *na-re*
`cv.abl`	converb *las*
`cv.are`	converb *ta-re, nga-re*, etc.
`cv.ass`	converb *dang*
`cv.odd`	any other converbs
`cv.rung`	converb *rung*
`interj`	interjections
`n.v.fut.n.v.past`	either fut. or past verb noun
`n.v.fut.n.v.pres`	either fut. or pres. verb noun
`n.v.imp`	nominalised imperative stem

Replaced Tags	Denotation
n.v	verb noun
n	noun

Additional Tags	Denotation
n.v.invar	past, pres. or fut. verb noun
n.v.pres	nominalised present stem
skt	non-words, often Sanskrit
v.fut.v.past	either fut. or past verb stem
v.fut.v.pres	either fut. or pres. verb stem
v.invar	past, pres. or fut. verb stem
v.past.v.pres	either past or pres. verb stem

Table 3. Changes in POS tag set compared to Garrett et al. (2015)

The cl-prefixes were replaced by d-prefixes, because the morphemes in question appear to be mostly restricted to noun phrases. The n.v.redup tag was removed, because it was realised that reduplicated structures were easily extractible from the corpus ex post facto, and that the tag n.v.redup was destroying the information about the tense of the reduplicated stem. Some tags were further specified, like v (verb), n.v (verb noun) and n (noun). Further information on tense and, for nouns, on number (e.g. count vs. mass) was added. New additions include the aforementioned further specifications for verb and verb nouns in terms of past, present and future tense (or an invariable category for those forms for which it is impossible to decide which tense it is). Another range of additional tags include more converbs (e.g. to distinguish ལས་ *las*, རུང་ *ruṅ* and དང་ *daṅ*) and one additional case marker ནའོ་ *na-re*, which introduces direct speech.[9] Mimetic adverbs such as ཅེའོ་ *ce re* 'jealous, wide-eyed' or ནར་ནར་ *nar nar* 'in a line' are now tagged "adv.mim." Finally, interjections like སྐྱེ་མ་ *kye ma* 'alas' are now tagged "interj" and onomatopoeia and other words that do not exist in Tibetan (and are frequently *dhāraṇī* or the like from Sanskrit) are tagged "skt." The complete tag set with further explanation and examples can be found on http://larkpie.net/tibetancorpus/tags (accessed 15 March 2017).

The memory-based POS tagger was generated in exactly the same way as the above-described Tibetan syllable tagger with the Memory-Based Tagger generator by TiMBL (see Appendix A for further details). The parameter settings and frequency lists were automatically derived from the manually corrected training set consisting of Tibetan words (and/or case endings etc.). Again, different parameter settings and algorithms could be selected for known and unknown tokens. The optimal settings were found by systematically evaluating and comparing the results of the various combinations. The complete workflow is shown in Figure 8:

4 Results and evaluation

In order to evaluate the quality of the POS tagger and obtain optimal parameter settings, we evaluated on the manually corrected training data. We did so with a ten-fold cross-validation, i.e. taking 90% of the data, training the model on that subset and then testing it on the other 10%, repeating this procedure for ten 90%/10% splits. Because the ten percent the model is tested on is

9 Case markers are split from their immediately preceding (nominal) forms and tagged separately as "genitive, allative, agentive, associative, elative, ablative or terminative case", because doing so simplifies lemmatisation.

manually corrected, we can see how often the model assigns the correct tag to a word, as well as obtain insightful statistics about the over- and undergeneralisations of some tags. In addition to the tagger performance, we also report the results of both the syllable tagger and the POS tagger for each of the tags in terms of Precision (percentage of assigned tags that were correct), Recall (percentage of tags in the input that were correctly identified by the system) and F-score (weighted harmonic mean of Precision and Recall).

4.1 Word segmentation results: Syllable tagging + rule-based conversion

For the segmentation task, we tested the model with two slightly different tag sets: a small tag set (expected to get better results because there are fewer options for the tagger) and a tag set with two extended tags Y and Z for additional initial syllables. It is not surprising that the optimal parameter settings for each of the tag sets do not differ much, since the tag sets themselves only slightly differ from each other. The additional Y and Z tags in the larger tag furthermore do not occur very frequently in comparison to the absolute initial syllable X. The manually corrected training set is quite large: there are plenty of examples for each of the different tags in context. Tables 4 and 5 show a complete overview of the results for each of the syllable tags (both for known and unknown syllables) for the small and the larger tag sets respectively, including Precision, Recall and F-scores as well as overall frequencies of occurrence.

Tag	**Precision**	**Recall**	**F-score**	**Frequency**
S	0.96	0.96	0.96	202629
B	0.91	0.92	0.91	74911
E	0.9	0.9	0.9	53530
ES	0.97	0.97	0.97	21489
M	0.68	0.56	0.62	13544
SS	0.94	0.95	0.95	8521

Table 4.1. Results per tag for known syllables with the small tag set

Tag	**Precision**	**Recall**	**F-score**	**Frequency**
S	0.7	0.75	0.72	949
B	0.74	0.73	0.73	631
E	0.53	0.45	0.49	325
ES	0.48	0.44	0.46	198
SS	0.32	0.39	0.35	190
M	0.49	0.39	0.43	176

Table 4.2. Results per tag for unknown syllables with the small tag set

Tag	**Precision**	**Recall**	**F-score**	**Frequency**
S	0.96	0.96	0.96	202629
X	0.91	0.92	0.91	74911
E	0.9	0.89	0.9	53530
ES	0.97	0.97	0.97	21489
Y	0.66	0.61	0.63	8525
SS	0.94	0.95	0.94	8521
Z	0.56	0.43	0.49	3366
M	0.37	0.13	0.19	1653

Table 5.1. Results per tag for known syllables with the larger tag set (in which B = X, Y or Z)

Tag	**Precision**	**Recall**	**F-score**	**Frequency**
S	0.71	0.76	0.73	949
X	0.75	0.73	0.74	631
E	0.52	0.43	0.47	325
ES	0.45	0.41	0.43	198
SS	0.32	0.39	0.35	190
Y	0.39	0.5	0.44	88
Z	0.4	0.27	0.32	59
M	0.25	0.07	0.11	29

Table 5.2. Results per tag for unknown syllables with the larger tag set (in which B = X, Y or Z)

As usual, the MBT performs worse for the unknown tokens (i.e. the syllables that the tagger encountered that are not in the frequency list based on the training set) than for the known tokens. In general, these have a lower frequency as well, and it is obvious that the lowest frequencies (tags M and Z with n=29 and n=59 respectively) receive worse results.

Syllables in the middle of the word (with additional beginning and end syllables) and syllables following two more syllables at the beginning of the word are rare in Classical Tibetan. Therefore, both Precision and Recall are very low. Looking at the results for the most frequently found type of syllable, a single-syllable word (S), with over 180*k* tokens, Precision and Recall are 96% for known syllables (with n>202*k*) and 73% for unknown syllables (where n=949). The Global Accuracies for known and unknown syllables with both tag sets and the overall Global Accuracy are presented in Table 6:

	Small tag set	**Larger tag set**
Overall Global Accuracy	93.0%	92.7%
Global Accuracy known syllables	93.2%	92.9%
Global Accuracy unknown syllables	62.5%	62.3%

Table 6. Comparison of Global Accuracies for the di

The overall results are slightly better with the small tag set, which is not surprising, because the tagger has fewer options to choose from and thus makes fewer mistakes. This is mainly the case for unknown syllables where there is a difference of nearly 2% in performance of the tagger. The overall Global Accuracy of 93% for both is very good considering the minimal amount of preprocessing that was done.[10] If the syllables are assigned the incorrect tag, however, the rules of combining syllables into proper Classical Tibetan words or case endings cannot be predicted. Therefore, the rule-based script that finalises the word segmentation task may yield the wrong results. Take for example the following six consecutive syllables presented to the tagger that were particularly challenging: ཆུ་ བོ་ འོད་ སྲུང་ སྤུན་. The correct tags for this particular sequence of tags would be: ཆུ་/X བོ་/Y འོད་/Z སྲུང་/M སྤུན་/E. Based on the training set and the parameter settings the tagger starts assigning the correct tag to the first syllable: ཆུ་/X. Because the syllable བོ་ often receives an E tag in the training set (e.g. in sequences like སྐྱེ་ བོ་, tagged སྐྱེ་/X བོ་/E, to be combined into སྐྱེ་བོ་ *skye po* 'human being'), the tagger has now also assigned an E tag to the second syllable: བོ་/E. If the output of the syllable tagger is not manually corrected, the rule-based script will now combine the first two syllables together, because sequences of X and E will automatically be combined: ཆུ་/X བོ་/E > ཆུ་བོ་. The rest of the syllables are now also assigned the incorrect tags, because after a syllable tagged E (indicating the end of the word), the next syllable should indicate the start of a new word tagged by X (instead of the correct Z, which is the syllable following X- and Y-syllables respectively). This means that the rule-based script will now combine these following syllables into a new word instead of combining it all into the proper sequence ཆུ་བོ་འོད་སྲུང་སྤུན་ *chu bo 'od srung spun*, a personal name. Table 7 gives an overview of the correct and incorrectly assigned tags in this particular sequence of syllables:

Syllable sequence	**Correct tag**	**MBT-assigned tag**
ཆུ་	X	X
བོ་	Y	E
འོད་	Z	X

10 Note that this percentage cannot easily be compared to the reported 95.12% in Huidan et al 2011, as they only report the F-score of the CRF word segmenter for Tibetan (CRF-SegT) based on their relatively large training set of >131*k* tokens with a small test set of 1,000 sentences. Our second stage of combining the tagged syllables into words presumably is incorporated in their evaluation metric. For the present paper, we only focussed on evaluating the results of the memory-based syllable and POS taggers. We leave a further comparison between the CRF-SegT and our approach, also in terms of how complex and time-consuming pre- and post-processing tasks are for future research.

སྐྱོང་	M	E
སྐྱེལ་	E	S
ལ་	S	S

Table 7. Sample of incorrectly assigned tags

Especially because establishing the correct tags for the left and right context is of crucial importance in sequences of syllables, most errors occur in clusters as in the above example: when one syllable is assigned a tag indicating the end of the word rather than a sequence of syllables in the beginning, the following syllables are often tagged incorrectly as well. This happens in particular with syllables that can occur in multiple positions in the word and have thus received more than one possible tag in the training set.

If further information about Tibetan syllable structure and/or typical word length were added to the model (e.g. in the form of complex feature templates), the results would probably improve. However, the design of such feature templates would take time and complicate the present workflow. 93% Accuracy is a sufficiently good result for subsequent POS tagging, after which manual correction will have to be done anyway before undertaking further NLP tasks like (chunk)parsing. The results of the automatic syllable tagging + rule-based conversion can thus serve as direct input for the POS tagger.

4.2 POS tagging results

The tag set for morphosyntactic tagging is reasonably large (79 tags), but the overall results of the tagger are still good. Punctuation markers are tagged as well and occur very often in the training set in the exact same form, which is why the scores for these tags are so high. Genitive and agentive case markers are also easily retrieved by the tagger, and the Precision of these tags is high as well, because they have a very recognisable form. Count nouns, on the other hand, appear in many shapes and forms, but with an F-score of 96% the tagger was able to assign this tag to nouns and retrieve them correctly most of the time, because of the high frequency of the tag "n.count" and because there are not many different tags assigned to nouns in general (only n.count, n.mass, n.prop and n.rel).

For verbs, there are numerous tags to indicate different tenses etc. Unsurprisingly, the results for the tag "v.invar" are not as good (F-score of 88%), because this tag is used for verb stems that are morphologically and syntactically ambiguous: they could be either present, past or future, e.g. གཤེགས་ *gshegs* 'to depart' in the phrase གཤེགས་སོ་ *gshegs so* 'departs/departed'. Tables 8 and 9 show the results of the most-frequent tags.

Category	Precision	Recall	F-score	Frequency
n.count	0.96	0.96	0.96	47802
punc	1	1	1	42353
case.gen	0.99	1	1	17496
case.term	0.98	0.99	0.99	15745
case.agn	0.98	1	0.99	11247
v.past	0.95	0.96	0.96	10351
n.v.invar	0.97	0.97	0.97	9134
d.dem	0.99	0.98	0.99	9088
case.all	0.98	0.99	0.99	7960
v.invar	0.89	0.87	0.88	7935

Table 8. Results for 10 most-frequent tags for known words

Category	Precision	Recall	F-score	Frequency
n.count	0.61	0.84	0.7	6048
n.prop	0.57	0.38	0.46	3424
adj	0.23	0.15	0.18	578
n.v.invar	0.14	0.13	0.13	339
n.v.past	0.29	0.33	0.31	300
n.v.fut.n.v.pres	0.21	0.17	0.19	279
v.invar	0.26	0.27	0.27	241
v.past	0.4	0.6	0.48	238
n.v.pres	0.12	0.04	0.06	178
dunno	0	0	0	168

Table 9. Results for 10 most-frequent tags for unknown words

The F-scores for unknown words are much lower, as always, but some categories still perform reasonably well, like the tags “n.count” and “v.past.” As Classical Tibetan does not have distinctive morphology for adjectives, this category too is expected to get a low Recall rate (15%). The same is true for the “dunno” category, which literally indicates that even human taggers found it impossible to decide to which morphosyntactic category the word belongs. A sample of the resulting POS-tagged file is shown in Figure 9. Again, known tokens are indicated by a single /, unknown tokens are indicated by a double //.

```
རོལ་མོ/n.count དང་/case.ass དམ་པ/n.v.invar དེ/case.gen ཚ་བྱད/n.count ཐམས་ཅད/d.plural
རང/d.det ཡོད་པ/n.v.invar ར/case.term ཐུར/v.imp ཅེས/cv.imp ཅེས/cl.quot སྨོས/v.past
མ/neg ཐག/v.aux ཏུ/cv.term ཤིང/n.count ཐམས་ཅད/d.plural ཀྱི/case.gen ཡལ་གའ/n.count
ལས/case.abl རོལ་མོ/n.count དེ/case.gen ཚ་བྱད/n.count མང་པོ/adj འཕྲང//v.fut.v.pres
ཤེ/cv.sem |/punc <utt>
རོལ་མོ/n.count འདོད་པ/n.v.invar ཞེག/d.indef ཡོད/v.invar ན/cv.loc |/punc <utt>
རང/p.refl འཚེལ/v.fut.v.pres ཞིང/cv.impf སྣན་པ/n.v.pres དང/case.ass |/punc <utt>
ཐོས་པ/n.v.invar ཙམ/d.tsam ཀྱིས/case.agn དགའ/v.invar རངས་པ/n.v.invar ར/case.term
གྱུར/v.past ཏེ/cv.fin |/punc <utt>
རྒྱལ་པོ/n.count དེ/case.gen བསོད་ནམས/n.count དང/case.ass དབང་ཐང/n.count ཆེན་པོ/adj
པོ/case.agn ནམ་མཁའ/n.count ལས/case.abl རིན་པོ་ཆེ/n.mass སྣ/n.count བདུན/num.card
ཆར/n.count བཞིན/n.rel དུ/case.term བབས/v.past ནས/cv.ela |/punc <utt>
ཤུལ/n.count ཀུན/d.plural གང/p.interrog ནས/case.ela རྒྱལ་པོ/n.count ས/case.agn བློན་
པོ/n.count དག/d.plural ལ/case.all འདི/d.dem ཐུ་བོ/n.rel ར/case.term གྱུར་པ/n.v.past
འདི/d.dem སུ/p.interrog དེ/case.gen ཡོན་ཏན/n.count ཞེས/cl.quot 285b//numeral དྲུག
པ/n.v.past དང/case.ass |/punc <utt>
བློན་པོ/n.count རྣམས/d.plural ཀྱིས/case.agn འདི/d.dem སྐད/n.rel ཅེས/cl.quot སྨོས/v.past
ཤིག/cv.fin |/punc <utt>
```

Figure 9. Sample of POS-tagged file.

The training set is large and the total number of unknown words is comparatively small. In addition, the results for the known words are very high (Global Accuracy of 96.8%). Therefore, these lower results for unknown words do not significantly lower the overall Global Accuracy:

	Results MBT with 79 morphosyntactic tags
Overall Global Accuracy	95.0%
Global Accuracy known words	96.8%
Global Accuracy unknown words	53.4%

Table 10. Global Accuracies of POS tagging with the generated MBT (micro-averaged results)[11]

Looking at the errors, there are various cases of proper nouns (n.prop), that were tagged as count nouns instead (n.count), such as ཤཱཀྱ་སེང་གེ་ *śākya seng ge* 'lion of the Śakyas'. This also happens the other way around, where a count noun is tagged as a proper noun, e.g. བསྟོད་སྨད་ *bstod smad* 'praise and blame'. The lack of capital letters to discriminate proper nouns from count nouns in the Tibetan script makes it more difficult for the tagger to recognise these instances. There are, however, plenty of occasions where unknown count nouns or proper nouns are recognised as such and correctly tagged "n.count" or "n.prop" respectively, e.g. དཀར་ཕྱོགས་ལྷ་ *dkar phyogs lha* 'good spirits' and བྱམས་པ་སེང་གེ་སྒྲ་ *byams pa seng ge sgra*, a proper name.

On some occasions, unknown adjectives are tagged as count nouns instead, e.g. འདབ་སྟོང་ *'dab stong* 'thousand-petalled' or མཆོག *mchog* 'supreme'. Again, the reverse occurs as well, e.g. སྙིང་པོ *snying po* 'heart'. Since there is no distinct morphology for adjectives, these are difficult to disambiguate for the tagger. Often the context helps, e.g. ཡེ་ཤེས་ཐེག་པ་ *ye shes theg pa* 'wisdom vehicle' (n.count) + མཆོག་འགྱུར་ *mchog gyur* 'supreme' (adj). Rather than tagging two consecutive words as count nouns, the tagger correctly decides to tag the second word as an adjective, even though it has not encountered this particular word in the training set.

[11] These results are based on gold standard syllable structured data. In future work we hope to address the combined results based on the syllable tagger output as well (see Hill and Meelen forthcoming).

In the verbal domain, similar errors occur, especially with verb tags that indicate multiple options, such as "v.fut.v.past" for verbs that are either future or past tense or "n.v.fut.n.v.pres." for verbal nouns that are either future or present. The tagger can easily analyse those as one or the other instead of the correct combined tag. སྤྱད་ *spyad* 'practised' was tagged as "v.past", but should have been tagged as "v.fut.v.past". Similarly, བསལ་བ་ *bsal ba* 'will remove' was tagged as "n.v.fut.n.v.pres", but should have been tagged as "n.v.fut". Needless to say, without the help of inflectional prefixes or endings or the context to distinguish between the two tags, it is very difficult for the tagger to predict the correct result.

5 Conclusion and future application

In this paper we presented a new approach to word segmentation and Part-of-Speech tagging in Classical Tibetan. The workflow consists of generating a Memory-Based Tagger that first of all assigns location tags to syllables of a minimally preprocessed new set of data. After this memory-based syllable tagging, we use a small rule-based script to convert the tagged syllables into proper Tibetan words. We then employed the same memory-based software by TiMBL to generate a POS tagger that assigns morphosyntactic tags to the segmented text files. We finally explored the optimal parameter settings for different tag sets for the syllable tagger and for the POS tagger and evaluated the results of both stages with a 10-fold cross-validation that yielded Global Accuracies of 93% and 95% for syllable and POS tagging respectively (see Appendix A for further details on Memory-Based Tagging and Appendix B for an exhaustive list of results).

The proposed staged workflow has several advantages. First of all, the results of both challenging NLP tasks are good, especially if we take certain difficulties into account concerning the Tibetan script (e.g. the lack of capital letters) and language (e.g. limited morphological suffixes to disambiguate certain categories). Secondly, these results were achieved with a minimal amount of preprocessing. This means that for future applications, digitised Tibetan text can be fed into the syllable tagger by merely adding utterance boundaries following ། *shad* and inserting spaces after ་ *tsheg*. No further rules, feature templates, editorial changes or any prior knowledge of Tibetan is necessary to employ the proposed workflow on a large scale. Finally, each stage of the workflow can be performed in a matter of seconds. We therefore conclude that this workflow would be a very suitable way of creating a large annotated corpus of Classical Tibetan.

6 Appendix A: Memory-Based Tagging and Parameter settings

For the present paper we used the Memory-Based Tagger (MBT) and MBT generator provided by TiMBL because it is easy to use, it does not require time-consuming preprocessing and it yielded good results (see Appendix B for a full list). In this Appendix, we provide some further technical background to the process, starting with a short introduction to the tagger, followed by a description of the parameter settings and finally, an overview of the optimal parameter settings used for the current corpus.

6.1 Memory-Based Tagging

Memory-Based Tagging is an approach to sequence tagging based on *Memory-Based Learning* (MBL) going back to the 1990s (cf. Daelemans 1995). It adapts and extends the classical k-Nearest Neighbor (k-NN) approach to statistical pattern classification. As such, MBL has proven to be quite successful in a wide variety of tasks in Natural Language Processing (Daelemans and Van den Bosch, 2005). The Memory-Based Tagger (MBT) based on TiMBL software yields good results for POS tagging because it uses previous tagger decisions as input for current decisions, builds separate case bases for known and unknown words and allows for global sentence-level optimization. The most recent version of software and documentation are available under the GNU General Public License from http://ilk.uvt.nl/timbl.

6.2 Optimising parameter settings

In order to optimise the results, there are numerous different parameters, each with their own range of settings, available for the MBT. The default settings of the generated tagger are:

`-p ddfa` (for known tokens)
`-P dFapsss` (for unknown tokens)

The flags -p and -P indicate the settings for known and unknown tokens respectively. "d" and "a" refer to the direct context on the left and on the right of the focus word ("f"/"F") respectively. Instead of specifying whether the tagger needs to take the left or right context into account, it can do either with the additional setting "w." For unknown tokens, there are additional features for characters at the beginning ("p") or end ("s") of the word. In addition, the tagger could be set to take further features like capitalised characters ("c"), hyphens ("h") and numerical characters ("n") into account. As these are not part of the Tibetan script, however, they were not used to generate the Tibetan syllable and POS taggers.

With these parameter settings for known and unknown tokens in place, the Memory-Based Tagger generator (MBTg) first creates a list of tokens that are associated with all the tags it can have in the training corpus. Whenever a token-tag combination occurs less than 5% (by default, this too is an adjustable setting), it is not included. After this, the MBTg creates a frequency list of the 100 (again, by default, but this number is adjustable too) most frequent tokens in the corpus. If a token is not found in this frequency list, it will be transformed into a special symbol: HAPAX-<code> (<code> is either 0, or a combination of H (hyphen), C (capital letter), and N (number)). On the basis of the feature settings for known and unknown tokens above, the case base for known words is generated by TiMBL. By default, a lazy-learning algorithm like IGTREE is used (cf. Daelemans and Van den Bosch (2005)). However, for both the syllable as well as the POS tagging tasks in Classical Tibetan, the IB1 algorithm rendered better results for both known and unknown words. This is a *k*-nearest-neighbour algorithm (based on Aha et al. (1991) but with added *Information Gain* weighting, cf. Daelemans and Van den Bosch (2005)). The selected feature metric was set to a Modified Value Difference Metric (MDVD) with the -mM parameter. This metric allows for partial feature matches (cf. Stanfill and Waltz (1986), Cost and Salzberg (1993) and Daelemans and Van den Bosch (2005)). Finally, weighting of features can be done in different orders. For both syllable and POS tagging, the inverse linear fashion with the parameter setting -dIL rendered the best results. This means that the

neighbour with the smaller distance is weighted more heavily than the one with a greater distance. From all these parameter settings and input from the training set, a settings file is created that can then be used to annotate any new texts in the future.

To find the optimal parameter settings for the present training corpora (both for segmentation and POS tagging), we ran dozens of 10-fold cross-validations on the datasets: each of these 10-fold cross-validations was tested with a specific combination of parameter settings and the results were then systematically compared to tests with different settings. We started with the parameter settings that were most optimal in another complex and minimally preprocessed dataset, i.e. the Middle Welsh native prose corpus (cf. Meelen 2016), changing one variable at a time and comparing the results to previous tests thus improving the combinations of settings and ultimately the overall results.[12] The following parameter settings gave the optimal results for the small and larger tag sets used for syllable tagging respectively:

	Small tag set	**Larger tag set**
Features known syllables	`-p dwdwdwfWawaw`	`-p dwdwdwfWawaw`
Features unknown syllables	`-P psssdwdwFawaw`	`-P psssdwdwFawaw`
Frequency & ambitag lists	`-M 1100 -n 3 -% 5`	`-M 900 -n 3 -% 5`
Algorithm known syllables	`-a0 -w2 -k1`	`-a0 -w2 -k1`
Algorithm unknown syllables	`-a0 -w2 -mM -k1 -dIL`	`-a0 -w2 -mM -k1 -dIL`

Table 11. Optimal Parameter settings for small and larger tag sets

As is clear from Table 11 above, the optimal settings for the feature patterns for known and unknown syllables are the same in both the small and the larger tag set. Recall from section 3.2 above that these include features such as the characters at the beginning and end of the syllable (p and s) as well as the left and right context (a and w). Unsurprisingly, the special features for hyphens (h) or capitalised characters (c) did not help the tagger as these do not exist in the Tibetan script. There is a slight difference in the total number of words in the list of the most frequently found syllables in the corpus. The default for this setting is 100, but larger lists of 1100 and 900 syllables for the small and larger tag sets respectively (modified by -M) yielded better results. The option -n determines the maximum frequency a syllable can have in the training corpus to use it in its context in the creation of the unknown words case base. The default value is 5, but a lower value of 3 rendered better results for the small tag set. The threshold of 5% of the syllable's total frequency for the ambitag lexicon was not changed as it did not improve the results in any way. The algorithm that rendered the best results for both tag sets is IB1 (the above-described *k*-Nearest-Neighbour algorithm (selected by -a0) with Information Gain weighting (selected by -w2)). The preferred distance metrics for each feature for both tag sets was the Modified Value difference (-mM). The option -k indicates the number of nearest neighbours that is taken into account. Here, the default number 1 yielded the best results for

12 In future research, we hope to test all settings automatically, but since there technically are millions of options combining all the different settings (including varying values, e.g. for the size of the lexicon, feature distance, etc), this can only done systematically with enough computing power and an appropriate script automating each 10-fold cross-validation and listing the results.

both the smaller and the larger tag sets in known and unknown syllables. Finally, for both tag sets, the Inverse Linear class type was the best in terms of voting weights used for the extrapolation from the nearest neighbour set.

For the large POS-tag set, the optimal parameter settings were tested in a similar fashion with a series of 10-fold cross-validations. The optimal settings on the POS training set were as follows:

Features	**Parameter settings**
Features known syllables	`-p dwdwfWaw`
Features unknown syllables	`-P psssdwdwdwFawaw`
Frequency & ambitag lists	`-M 1100 -n 5 -% 8`
Algorithm known syllables	`-a0`
Algorithm unknown syllables	`-a0 -mM -k17 -dIL`

Table 12. Optimal parameter settings for the POS tagger

These settings are very similar to the ones used for syllable tagging above. Again the IB1 algorithm is used (-a0). The number of nearest neighbours (-k17) is slightly higher, but the Inverse Linear class type for voting weights works best again. A large frequency list (-M1100) gave the best results, as did the features to marker characters at the beginning (p) and end (s) of the word, along with those for left and right contexts (a and w).

It is important to note that the better-performing IB1 algorithm is with ~500 tokens/second considerably slower than its IGTREE alternative, which can deal with 1000s of tokens per second for this particular corpus. The default settings of the MBT are therefore a combination of IGTREE and IB1. For segmenting or tagging a single file this is no problem, but for the 300*m* corpus of Classical Tibetan, this poses a significant challenge (see Hill and Meelen forthcoming).

7 Appendix B: Complete evaluation per tag for POS tagging for known and unknown words

Category	**Precision**	**Recall**	**F-score**	**Frequency**
`n.count`	0.96	0.96	0.96	47802
`punc`	1	1	1	42353
`case.gen`	0.99	1	1	17496
`case.term`	0.98	0.99	0.99	15745
`case.agn`	0.98	1	0.99	11247
`v.past`	0.95	0.96	0.96	10351

n.v.invar	0.97	0.97	0.97	9134
d.dem	0.99	0.98	0.99	9088
case.all	0.98	0.99	0.99	7960
v.invar	0.89	0.87	0.88	7935
n.rel	0.93	0.96	0.94	7029
case.ass	0.99	1	0.99	6947
n.v.past	0.97	0.97	0.97	6674
num.card	0.99	0.99	0.99	6164
cv.fin	0.99	1	1	5984
cl.focus	0.98	0.99	0.98	5766
cv.sem	0.99	0.99	0.99	5553
d.plural	0.99	0.99	0.99	4762
neg	0.98	0.99	0.98	4328
p.pers	0.98	0.98	0.98	4248
cv.ela	0.97	0.98	0.97	4030
v.fut.v.pres	0.88	0.88	0.88	3883
adj	0.93	0.94	0.93	3777
n.prop	0.94	0.92	0.93	3776
n.v.fut.n.v.pres	0.93	0.94	0.93	3445
cl.quot	1	1	1	3098
n.v.pres	0.96	0.95	0.96	2936
v.pres	0.95	0.93	0.94	2533
cv.term	0.95	0.89	0.92	2425
case.ela	0.96	0.93	0.94	2082
d.indef	0.98	0.98	0.98	2005
p.interrog	0.97	0.96	0.96	1929
case.loc	0.93	0.95	0.94	1921
v.past.v.pres	0.85	0.91	0.88	1787
v.imp	0.84	0.75	0.79	1700
cv.loc	0.94	0.92	0.93	1679
adv.temp	0.93	0.92	0.93	1653
n.v.past.n.v.pres	0.95	0.96	0.95	1651

n.mass	0.9	0.92	0.91	1617
v.cop	1	1	1	1591
cv.impf	0.98	0.99	0.98	1562
n.v.fut	0.99	0.98	0.98	1379
v.fut.v.past	0.9	0.89	0.9	1377
case.abl	0.97	0.99	0.98	1335
v.fut	0.93	0.9	0.92	1172
d.det	0.92	0.9	0.91	1155
adv.intense	0.94	0.96	0.95	918
adv.proclausal	0.92	0.89	0.9	875
cv.imp	0.96	0.96	0.96	824
n.v.neg	1	1	1	778
cv.all	0.9	0.74	0.81	660
num.ord	0.99	0.99	0.99	654
cv.ques	0.97	0.97	0.97	651
n.v.fut.n.v.past	0.96	0.98	0.97	627
v.neg	1	1	1	617
p.refl	0.89	0.9	0.89	585
v.aux	0.82	0.82	0.82	512
d.tsam	0.98	0.98	0.98	479
p.indef	0.9	0.93	0.91	391
d.emph	0.95	0.9	0.93	351
n.v.cop	1	0.99	1	346
cv.agn	0.86	0.71	0.78	331
cv.gen	0.76	0.61	0.68	201
cv.rung	0.9	0.95	0.93	194
dunno	0.26	0.15	0.19	182
case.nare	1	0.99	1	174
cv.ass	0.71	0.49	0.58	157
n.v.aux	0.66	0.73	0.69	150
case.comp	0.99	0.99	0.99	135
v.cop.neg	1	1	1	123

adv.dir	0.91	0.88	0.9	122
interj	0.94	0.83	0.88	35
skt	0.52	0.38	0.44	32
cv.cont	0.96	0.96	0.96	27
cv.odd	0.81	0.85	0.83	26
n.v.imp	1	0.92	0.96	13
cv.abl	0	0	0	4
adv.mim	0.33	0.5	0.4	4
cv.are	1	1	1	4

Table 13.1. Complete results of POS tagging for known words

Category	**Precision**	**Recall**	**F-score**	**Frequency**
n.count	0.61	0.84	0.7	6048
n.prop	0.57	0.38	0.46	3424
adj	0.23	0.15	0.18	578
n.v.invar	0.14	0.13	0.13	339
n.v.past	0.29	0.33	0.31	300
n.v.fut.n.v.pres	0.21	0.17	0.19	279
v.invar	0.26	0.27	0.27	241
v.past	0.4	0.6	0.48	238
n.v.pres	0.12	0.04	0.06	178
dunno	0	0	0	168
v.fut.v.pres	0.35	0.24	0.28	149
n.v.fut	0.17	0.06	0.09	124
v.pres	0.05	0.03	0.04	101
n.mass	0	0	0	101
v.imp	0.38	0.23	0.28	88
n.v.past.n.v.pres	0	0	0	76
v.fut	0.35	0.09	0.14	69
skt	0.07	0.03	0.04	65

adv.temp	0	0	0	64
n.v.fut.n.v.past	0	0	0	55
v.fut.v.past	0.08	0.05	0.06	43
v.past.v.pres	0.16	0.12	0.14	42
num.card	0	0	0	27
n.rel	0	0	0	26
adv.mim	0	0	0	18
interj	0	0	0	15
d.dem	0	0	0	14
numeral	0	0	0	13
case.ass	0	0	0	12
num.ord	0.2	0.09	0.13	11
p.pers	0	0	0	11
n.v.imp	0	0	0	9
cl.focus	0.2	0.14	0.17	7
d.tsam	0	0	0	7
d.plural	0	0	0	6
cv.fin	1	0.33	0.5	6
adv.intense	0	0	0	5
cl.quot	0	0	0	3
cv.ques	0	0	0	3
adv.proclausal	0	0	0	2
cv.imp	0	0	0	2
adv.dir	0	0	0	2
case.gen	0	0	0	2
d.det	0	0	0	2
n.v.cop	0	0	0	1
cv.impf	0	0	0	1
n.v.neg	0	0	0	1
p.interrog	0	0	0	1
p.refl	0	0	0	1
d.indef	0	0	0	1

`cv.cont`	0	0	0	1
`case.comp`	0	0	0	1

Table 13.2. Complete results of POS tagging for unknown words

REFERENCES

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. "Instance-based learning algorithms". *Machine Learning* 6.1: 37-66.

Cost, S.; and Salzberg, S. 1993. "A weighted nearest neighbor algorithm for learning with symbolic features". *Machine Learning* 10.1: 57-78.

Daelemans, W. 1995. "Memory-based lexical acquisition and processing". In P. Steffens (Ed.), *Machine translation and the lexicon*, volume 898 of *Lecture notes in artificial intelligence*, 85–98. Berlin: Springer-Verlag.

Daelemans, W.; and Van den Bosch, A. 2005. *Memory-based language processing.* Cambridge, UK: Cambridge University Press.

Daelemans, W.; Zavrel, J.; Berck, P.; and Gillis, S. 1996. "MBT: A memory-based part of speech tagger-generator". In: E. Ejerhed and I. Dagan (Eds.), *Proceedings of the Fourth Workshop on Very Large Corpora*, 14-27. Copenhagen.

Daelemans, W.; Zavrel, J.; Van den Bosch, A.; and Van der Sloot, K. 2010. *MBT: Memory-Based Tagger, version 3.2, Reference guide*. Tilburg: Tilburg University [ILK Technical Report Series 10-04].

Garrett, E.; Hill, N. W.; and Zadoks, A. 2014. "A rule-based part-of-speech tagger for Classical Tibetan". *Himalayan Linguistics* 13.2: 9-57.

Garrett, E.; Hill, N.W.; Kilgarriff, A.; Vadlapudi, R.; and Zadoks, A. 2015. "The contribution of corpus linguistics to lexicography and the future of Tibetan dictionaries". *Revue d'etudes Tibétaines* 32: 51-86.

Hill, Nathan W. 2012. "Tibetan *-las*, *-nas*, and *-bas*." *Cahiers de Linguistique Asie Orientale* 41.1: 3-38.

Hill, Nathan W. and Meelen, Marieke (In preparation) "Building an annotated corpus of Classical Tibetan (ACTib)".

Huidan, L.; Nuo, M.; Ma, L. L.; Wu, J.; and He, Y. 2011. "Tibetan word segmentation as syllable tagging using conditional random field". In: *Proceedings of the Pacific Asia Conference on Language, Information and Computation* (25), 168-177.

Meelen, Marieke 2016. *Why Jesus and Job spoke bad Welsh: The origin and distribution of V2 orders in Middle Welsh*. Utrecht: LOT publications.

Stanfill, C.; and Waltz, D. 1986. "Toward memory-based reasoning". *Communications of the ACM* 29.12: 1213-1228.

Zavrel, J.; and Daelemans, W. 1997. "Memory-based learning: Using similarity for smoothing" In: *Proceedings of the 35th annual meeting of the Association for Computational Linguistics and eighth conference of the European chapter of the Association for Computational Linguistics,* 436-443. Association for Computational Linguistics.

Zhao, H.; Huang, C.; Mu L.; and Lu, B. 2006. "Effective tag set selection in Chinese word segmentation via conditional random field modelling". In: *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*, 87-94. Wuhan.

Marieke Meelen
mm986@cam.ac.uk

Nathan W. Hill
nh36@soas.ac.uk